Database Assignment

### Purpose and End User of my database

| My database is a database to store games for a new game sales company. It will sell a variety of games and will compete with the store already on xbox. It will store orders from the user and will also store the user's ID. The end user will be the company who wants to add new games to the already existing database and store online orders if something were to go wrong with an order or a refund were requested. |
| --- |

### Describe at least 3 implications that are relevant to your database and its use by the end user and why they are important

| End User Implications When creating a database we will need to consider how the end user will use the database. Data will need to be added or modified easily so that employees can do it quickly. For example, an employee using the database may need to fix data that is incorrect and it will need to be easily modified. Or a customer may want to change their data in the website and the database should update without a fuss. This is important to consider so that the end user can easily use and modify our data. Privacy Implications When creating a database we need to consider the privacy of the data stored in the table. Certain data will need to be encrypted so that it can not be accessed by customers and that the private data cannot. For example, bank account details or addresses are private information and need to be encrypted to make sure the data cannot be leaked to hackers. The data must be stored safely. This is important as those details are private to the customer. Usability Implications When creating a database we must consider the usability of the data in the table. Data will need to be easily accessed by both the customer and the employee fixing problems. For example, when a customer makes a purchase the data needs to automatically store the purchase to avoid the employee having to do it. This will reduce the work an employee has to do. Another example is the ability to easily write a query for a certain data or product. This is important as usability of a table makes the data easier to navigate. |
| --- |

### Database Design- Your Entity Relationship Diagram.

|  |
| --- |

### Database Testing Table: SQL Statements

| **Purpose** | **SQL Statement** | **Result Success?** |
| --- | --- | --- |
| Receive important order information | SELECT sales\_id, user\_id, product\_id, name, price FROM PRODUCT  JOIN SALES ON PRODUCT.product\_id = SALES.product\_id  JOIN USER ON USER.user\_id = SALES.user\_id  ORDER BY sales\_id | Success |
| Receive available products | SELECT name, description, game\_rating FROM product  ORDER by name ASC | Success |
| Receive purchased games and user information | SELECT user\_id, first\_name, surname, email, sales\_id, sale\_date FROM sales  JOIN user ON user.user\_id = sales.user\_id | Success |
| Ordering by Price | SELECT name, price FROM product  ORDER BY price | Success |
| Game Information | SELECT name, description, game\_rating FROM product  ORDER by name ASC | Success |
| Order by Game Rating | SELECT name, description, game\_rating FROM product  ORDER BY game\_rating DESC | Success |
| Sort by User | SELECT \* FROM user  ORDER by user\_id | Success |

### Relevant Implications- Explain how your database addresses the relevant implications that you identified at the start.

| End User Implications Firstly my database considered both the customer and the developer. The 3 table design meant that a customer, for example, could only see the product tab as the customer should not have access to something such as the user table or sales table which would breach the privacy of other customers. The database itself would function well and the users who would be using it would be able to buy products and have registered purchase information to their name and the developers would be able to view this easily Privacy Implications The multiple table setup for my design means that people who are only meant to see a specific data set will only see that. Access to one table can be given and the privacy of another table can be kept. This means that important data such as the email address or phone number or even the price someone else has paid for a game can be kept hidden from the customer. Different views assigned to specific users such as the order by game rating to a customer who may be looking for the best reviewed games. Order information of many orders would go to a developer who may need to sort a problem but not to a user who could be dangerous with it. Usability Implications The database idea itself will be extremely usable as it can hold a lot of different data for different purposes. When the programme has been coded this design will be very easy to make sense of and therefore will be more usable. The use of primary and foreign keys from the entity diagram means that when creating a view in SQL they can be tied together accordingly and function well. An example of this can be seen in the final code. Having a permanent query means the query can be stored in the database file as a few. This allows for it to be directly referenced in python or just be copied across with a place to always come back and find it. This also allows for easy usability. |
| --- |

### Showcase:

| SQL Database Product Table    The product table was very simple in SQL. It listed 7 key things I thought was necessary for a customer to know about a game. The product\_id was a primary key as it could be referenced in the sales table without having to be bought up again.  Sales Table  Here we have the sale\_id saved as the primary key. The two foreign keys are bridged into this table and it also means that with a foregn key we can access any information from the table that the primary key came from. DATETIME was added to get an exact time for a purchase  User Table    This table refers to the user and the primary key found here is also linked into the sales table. It holds most of the data that a user would have stored. However I decided not to add any banking details as it would’ve been unnecessary. Functions in Code Here is an example of one of my functions:    This function shows the table of users and gives information in a table like style. This specific function was similar to the one first I developed for products. The first functions were simple add, update, delete and show. Each of them were independent but were at that point only programmed to work only with my product (v0 and v1). However I developed this further by creating 2 functions that encouperated most tables into themselves. This saved not only space but also simplified the main code.    Here I managed to develop the code further by encouperating a way to delete any item ID that came from any table. This was also similar for the update function. However the update function only worked with two of the tables (user and product) as the foreign keys made it very hard to change a foreign key from its primary location as well as its foreign location. I left this out as I didn’t know how to do it.  An expected output for one of my show functions would look something like this:    This output is from the user show function mentioned above. The table worked well to display clear and accurate data. This data was ordered by separating out the tuples by using an item (representing a row of data) and using a limit on the amount of characters. Final Body of Code   The final code was built up every time I modified something or added something in the functions. It started off with some very simple if and elif statements that would then allow a function to run. It then expanded into more options for viewing as I added show options for all 3 data types. The code then expanded further with the addition of the try and except built in functions. These allowed for code errors in the code to be avoided and instead tell you what the error was in simple words and it also opted for the program to let you try to input again without stopping. The final addition was to fulfill the end user and privacy implications. This was the addition of a status check. Although it was not encrypted by some form of password protection the idea is that if you are a user you can only see the products, and a developer could see everything and modify as much as my functions allow. This fulfilled the end user implications, one for the customer and one for the developer or store owner/employee. Testing and Interface   The interface(Shown above) meant that the user had to input very little, often just a single letter. I tried to make it nice and clear by using the \n function to create a new line while still being inside the same string or input. I would then space out a gap in between different inputs. The function that was shown had a table like design (shown under functions in code). This was a very clear and obvious design that allowed for easy data viewing.  I tested the code after making a few major changes. This might have been the addition of a new function or the modification of a non working one. Sometimes when testing the Try and Except would have to be commented out by using a hashtag to see what the problem with the sql editor had occurred or a simple grammatical error. Testing however was generally a smooth process and the try and except were perfect for eliminating the human error side of things. |
| --- |

All Code versions can be found in the Zip folder that has been submitted

**Teacher Checklists:**

**AS91879- Develop a digital outcome to manage data**

**Credits: 4**

**NZQA:** [https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91883.pd](https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91883.pdf)f

| Achieved- Develop a digital outcome to manage data | **Evidence** |  |
| --- | --- | --- |
| using appropriate tools and techniques to structure, organise, query and present data for a purpose and end user |  | ✓ |
| applying appropriate data integrity and testing procedures |  | ✓ |
| describing relevant implications. |  | ✓ |
| Merit- Develop an informed digital outcome to manage data |  |  |
| using information from testing procedures to improve the quality and functionality of the outcome | For the limited functionality yes. This could have been much better with a bit mor time taken over it. | ✓ |
| structuring, organising and querying the data logically | The functions are useful | ✓ |
| addressing relevant implications. | * NEVER ask the user for the database ID and insert it into the database!!! It is automatically created when you insert data this is extremely problematic- it just silently fails to insert the data. * This break functionality and usability. | ✓ |
| Excellence- Develop a refined digital outcome to manage data |  |  |
| iterative improvement throughout the development and testing process |  |  |
| presenting the data effectively for the purpose and to meet end-user requirements. | The tables relational functionality is not really complete. For E you should have used the sales table |  |

**Develop a computer program**

**Credits:** 4 (Internal)

**NZQA:** <http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2018/as91883.pdf>

| **Achieved**  **Develop a computer program** | **Evidence** |  |
| --- | --- | --- |
| Wrote a program that performs a specific task using a suitable programming language |  | ✓ |
| Set out the program code clearly |  | ✓ |
| Documented the program with comments |  | ✓ |
| Tested and debugged to ensure that it works on a sample of expected cases |  | ✓ |
| **Merit**  **Develop an informed computer program** |  |  |
| Documented the program with variable names and comments that describe code function and behaviour |  | ✓ |
| Following conventions of the chosen programming language |  | ✓ |
| Tested and debugged the program in an organised way to ensure it works on expected and relevant boundary cases | Meh. It doesn;t crash but it exits with no user feedback. It chooses a default table to add/delete from if you get the table wrong. Generally not polished so not an E | - |
| **Excellence**  **Develop a refined computer program** |  |  |
| Ensured the program is a well structured logical solution to the task |  |  |
| Making the program flexible and robust |  |  |
| Comprehensively tested and debugged the program |  |  |

Comments:

Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Achievement Standard.